EL996277061

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Packing Steiner Trees

Inventors:

Kamal Jain

Mohammad Mahdian

Mohammad R. Salavatipour

ATTORNEY'S DOCKET NO. MS1-1658US

[0001] The invention pertains to network data delivery.

BACKGROUND

[0002] Multicast is packet communication between a single sender and multiple specific receivers on a network. A multicast route can be represented as a tree rooted at the sender with a receiver at each leaf, and possibly some receivers on internal nodes. Multicast service imposes specific requirements for network implementation. For instance, whenever a destination address of a data packet is a multicast address, the data packet must be routed from the sender to all targeted set of maximum size receivers. Instead of transmitting packets from a sender to each receiver separately, conventional multicast route identification techniques attempt to share links and minimize resource consumption to locate a tree that reaches all receivers. To deliver as many data streams to as possible to requesting customers, data broadcaster (sender) generally attempt to identify many such trees to the requesting clients.

[0003] Unfortunately, existing techniques to approximate edge-disjoint trees including as many edge-disjoint communication paths (as many as substantially possible) from a sender to requisite receiver nodes in networks that include sender(s), receiver(s), and router(s), generally do not produce a substantially optimal number of such trees as data senders generally desire. An edge-disjoint tree includes at least one edge disjoint path, wherein edge-disjoint means that the edges used in this tree are not used in the other trees. Instead, many such edge-disjoint trees/paths are overlooked. As a consequence, better techniques to

approximate edge disjoint trees connecting senders to receivers in networks that include Steiner nodes, or routers, are desired. This would provide broadcasters with improved capability to distribute multiple disparate data streams, and offer receivers more viewing/listening choices.

SUMMARY

[0004] Systems and methods for packing Steiner trees are described. In one aspect, a set of Steiner trees and paths are generated from an undirected graph of vertices representing terminal and Steiner nodes. The Steiner trees and the paths are merged to produce linked and edge disjoint S-Steiner trees. If a subset S of the vertices is edge connected, then at minimum there are substantially $\alpha_{|S|}k$ edge-disjoint Steiner trees for S, wherein α_s is a sequence that tends to an asymptotic approximation factor of $|S|/4$ as S tends to infinity.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] In the figures, the left-most digit of a component reference number identifies the particular figure in which the component first appears.

[0006] Fig. 1 shows an exemplary computing environment within which systems and methods to pack Steiner trees may be implemented.

[0007] Fig. 2 shows further exemplary aspects of system memory of Fig. 1, including application programs and program data to locate a substantially maximum sized Steiner trees that can be used in many practical applications to connect to a same set of terminal vertices.

[0008] Fig. 3 shows an exemplary set of vertices of in input graph *G*. The exemplary vertices are used to demonstrate a shortcutting procedure that removes wasteful paths from a Steiner tree.

[0009] Fig. 4 shows a set of nodes (e.g., network nodes) for showing that there are a certain number of edge-disjoint *S*-Steiner trees in a graph as a function of a recurrence relation.

[0010] Fig. 5 shows an exemplary procedure to locate substantially maximum sized Steiner trees that can be used in many practical applications to connect to a same set of terminal vertices. In particular, Fig. 5 shows a procedure to produce linked and edge-disjoint S-Steiner trees for multicast of streaming media.

DETAILED DESCRIPTION

Overview

[0011] From a theoretical perspective, techniques for finding Steiner trees connecting all senders and receivers (network terminals) are associated with fundamental theorems in combinatorics. At one extreme, when there are only two terminals, a Steiner tree is just a path between the terminals, which can be identified using the well-known Menger theorem. At the other extreme, when all the network vertices are terminals (i.e., only sender and receiver network nodes—no Steiner nodes, a Steiner tree is just a spanning tree of the graph, for which a solution is provided by the classical Nash-Williams-Tutte theorem. In this implementation, a Steiner node is a router, which is a device or, in some cases,

software in a computer, that determines the next network point to which a packet should be forwarded toward its destination. The router is connected to at least two networks and decides which way to send each information packet based on its current understanding of the state of the networks to which it is connected. A router may be included as part of a network switch.

[0012] *Theorem 1*: Graph $G(V,E)$ contains k edge-disjoint spanning trees if and only if:

$$E_G(P) \geq k(t-1),$$

for every partition $P = \{V_1, \ldots, V_t\}$ of V into non-empty subsets, where $E_G(P)$ denotes the number of edges between distinct classes of P. Since the problem of finding k disjoint spanning trees in a graph is a special case of finding k disjoint bases of a matroid, Theorem 1 can be derived from Edmonds' matroid partition theorem.

[0013] The Menger and the Nash-Williams-Tutte theorems are max-min theorems that can be generalized into a single theorem which says that the maximum number of edge-disjoint Steiner trees is the same as the integer part of the minimum value of

$$\frac{E_G(P)}{|P|-1},$$

where the minimum is over the set of all partitions of the vertices of graph that include at least one terminal in each class, with $E_G(P)$ denoting the number of edges between distinct classes of P, and $|P|$ denoting the number of classes of P. This statement is not true if we are not in the extreme cases described above (e.g., see Kriesell , reference [1]).

[0014] For instance, using Theorem 1, if a graph G is $2k$-edge-connected then it has k edge-disjoint spanning trees. Kriesell [1] conjectured that this generalizes to Steiner trees, i.e., if a set S of vertices of G is $2k$-edge-connected then there is a set of k edge-disjoint S-Steiner trees in G. This conjecture is *still open*, even with $2k$ replaced by any constant multiple of k. Notice that the edge-connectivity of the set S is an upper bound on the maximum number of edge-disjoint S-Steiner trees. Thus, a constructive proof for the above conjecture would provide a *constant-factor approximation* algorithm for the Steiner tree packing problem. This means that the algorithm produces a constant fraction of trees as would a substantially optimal algorithm.

[0015] The special case in which $V-S$ is independent is considered by Kriesell [1] and Frank et al. [2]. A corollary of Theorem 1 is that for a graph $G(V,E)$ and $S \subseteq V$, if $V-S$ is an independent set and S is $k(k+1)$-edge-connected, then G contains k edge-disjoint S-Steiner trees. A stronger version of this appeared in [2], where they weaken the requirement for connectivity of S to $3k$-edge-connectedness. This means that between any two nodes in S there are at least 3k-edge disjoint paths. They also prove a generalization of Theorem 1 to hypergraphs.

[0016] For an arbitrary set S, Petingi and Rodriguez [3] give a lower bound for the number of edge-disjoint S-Steiner trees, by showing that: if S is k-edge-connected in G and $|S| \geq 2$, then G has at least

$$\left\lfloor \left(\tfrac{2}{3}\right)^{|V-S|} k/2 \right\rfloor$$

edge-disjoint S-Steiner trees. This means that their algorithm has performance guarantee of $\lfloor (\frac{2}{3})^{|V-S|}/2 \rfloor$. This performance is very much worse, and in most practical cases amounts to triviality, than what is presented in the following detailed description.

[0017] In contrast to existing techniques (e.g., as discussed by references [1] through [3]), systems and methods described below in reference to Figs. 1-5, pack Steiner trees to approximate a substantially maximum number of edge-disjoint subgraphs of a given graph G that connect a given set of required points S. Distinct from conventional approaches to generating Steiner trees, systems and methods of the invention combine a collection of edge-disjoint paths with a collection of edge-disjoint Steiner trees by modifying the paths and trees in several (polynomial) iterations. This iterative process to modify the Steiner trees with edge-disjoint paths is termed Steiner tree packing. In particular, if a subset S in the graph is k-edge-connected, then there are $\alpha_{|S|}k$ edge-disjoint Steiner trees 206 for S, where α_s is a sequence that tends to $\frac{4}{s}$ (an asymptotic approximation factor of $|S|/4$) as S tends to infinity.

[0018] The selected asymptotic approximation factor provides a sufficient condition for the existence of edge-disjoint Steiner trees in a graph in terms of the edge-connectivity of the graph. This condition is substantially optimal when S consists of three (3) points—sender and receiver nodes.

[0019] These and other aspects of the systems and methods to pack Steiner trees are now described in further detail.

Exemplary Operating Environment

[0020] Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable computing environment. Although not required, the invention is described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Program modules generally include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types.

[0021] Fig. 1 illustrates an example of a suitable computing environment 120 on which the subsequently described systems, apparatuses and methods to pack Steiner trees for multicast of streaming media may be implemented. Exemplary computing environment 120 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of systems and methods the described herein. Neither should computing environment 120 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in computing environment 120.

[0022] The methods and systems described herein are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable include, but are not limited to, including small form factor (e.g., hand-held, mobile, etc.) computing devices (e.g., mobile phones, personal digital assistants—PDAs, etc.), multi-

processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and/or so on. The invention is also practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0023] As shown in Fig. 1, computing environment 120 includes a general-purpose computing device in the form of a computer 130. The components of computer 130 may include one or more processors or processing units 132, a system memory 134, and a bus 136 that couples various system components including system memory 134 to processor 132. Bus 136 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. By way of example, and not limitation, such bus architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnects (PCI) bus also known as Mezzanine bus.

[0024] Computer 130 typically includes a variety of computer readable media. Such media may be any available media that is accessible by computer 130, and it includes both volatile and non-volatile media, removable and non-removable media. System memory 134 includes computer readable media in the form of volatile memory, such as random access memory (RAM) 138, and/or

non-volatile memory, such as read only memory (ROM) 140. A basic input/output system (BIOS) 142, containing the basic routines that help to transfer information between elements within computer 130, such as during start-up, is stored in ROM 140. RAM 138 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processor 132.

[0025] Computer 130 may further include other removable/non-removable, volatile/non-volatile computer storage media. For example, a hard disk drive 144 may be used for reading from and writing to a non-removable, non-volatile magnetic media (not shown), a magnetic disk drive 146 for reading from and writing to a removable, non-volatile magnetic disk 148 (e.g., a "floppy disk"), and an optical disk drive 150 for reading from or writing to a removable, non-volatile optical disk 152 such as a CD-ROM/R/RW, DVD-ROM/R/RW/+R/RAM or other optical media. Hard disk drive 144, magnetic disk drive 146 and optical disk drive 150 are each connected to bus 136 by one or more interfaces 154.

[0026] The drives and associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules, and other data for computer 130. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 148 and a removable optical disk 152, it are appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks,

random access memories (RAMs), read only memories (ROM), and the like, may also be used in the exemplary operating environment.

[0027] A number of program modules may be stored on the hard disk, magnetic disk 148, optical disk 152, ROM 140, or RAM 138, including, e.g., an operating system 158, one or more application programs 160 to pack Steiner trees for multicast of streaming media, other program modules 162, and associated program data 164.

[0028] A user may provide commands and information into computer 130 through input devices such as keyboard 166 and pointing device 168 (such as a "mouse"). Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, serial port, scanner, digital camera, etc. These and other input devices are connected to the processing unit 132 through a user input interface 170 that is coupled to bus 136, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

[0029] A monitor 172 or other type of display device is also connected to bus 136 via an interface, such as a video adapter 174. The monitor can be utilized, for example, to present a user interface (UI) associated with the described systems and methods to pack Steiner trees for multicast of streaming media. In addition to monitor 172, personal computers typically include other peripheral output devices (not shown), such as speakers and printers, which may be connected through output peripheral interface 175.

[0030] Computer 130 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 182. In such an implementation, the computer 130 may be a data receiver and/or a data sender network node. Remote computer 182 may include some or all of the elements and features described herein relative to computer 130. Logical connections include, for example, a local area network (LAN) 177 and a general wide area network (WAN) 179. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

[0031] When used in a LAN networking environment, computer 130 is connected to LAN 177 via network interface or adapter 186. When used in a WAN networking environment, the computer typically includes a modem 178 or other means for establishing communications over WAN 179. Modem 178, which may be internal or external, may be connected to system bus 136 via the user input interface 170 or other appropriate mechanism.

[0032] Depicted in Fig. 1, is a specific implementation of a WAN via the Internet. Here, computer 130 employs modem 178 to establish communications with at least one remote computer 182 via the Internet 180. In a networked environment, program modules depicted relative to computer 130, or portions thereof, may be stored in a remote memory storage device. Thus, e.g., as depicted in Fig. 1, remote application programs 189 may reside on a memory device of remote computer 182. The network connections shown and described are

exemplary. Thus, other means of establishing a communications link between the computing devices may be used.

Exemplary Application Programs and Data

[0033] Fig. 2 is a block diagram that shows further exemplary aspects of system memory 134 of Fig. 1, including application programs 160 and program data 164 to locate a substantially maximum number of linked and edge-disjoint Steiner trees that can be used in many practical applications to connect in different ways to a same set of terminal vertices. The definition of a terminal vertex is implementation dependent. For instance, in a networking scenario, terminal vertices include sender and receiver network nodes. In its full generality, the Steiner tree packing module 202 has many practical applications such as in data broadcasting, VLSI circuit design, etc, as described in greater detail below in the section titled "An Exemplary Procedure".

[0034] Application programs 160 include, for example, Steiner tree packing module 202 to analyze an undirected finite graph ("*G*") 204 of vertices (input data) to produce a substantially maximum number of linked, edge-disjoint, and packed ("LED&P") Steiner tree(s) 206 to approximate a substantially maximum number of edge-disjoint subgraphs (S-Steiner trees) of the given graph *G* that connect a given set of requisite points *S*.

[0035] Points *S* are implementation dependent. For instance, in a computer data server networking application, the data server may have some number of requesting receiver network nodes to which the server is to multicast streaming

data. In this networking example, the required points S are requesting receiver network nodes. In yet another example, the terminal vertices may be elements on a circuit for the exchange of electrical signals. These are only a couple of examples of different applications and possible definitions of a data point/element of S.

[0036] To generate the LED&P Steiner nodes 206, the Steiner tree packing module 202 generates a collection of paths 208 and edge-disjoint Steiner trees 210 by modifying paths between graph data points and Steiner trees in several (polynomial) iterations. If a subset S in the graph is k-edge-connected, there are $\alpha_{|S|}k$ edge-disjoint Steiner trees 206 for S, where α_s is a sequence that tends to $\frac{4}{s}$ (an asymptotic approximation factor of $|S|/4$) as S tends to infinity. This iterative process, generates, removes shared edges (i.e., shortcuts), and merges the edge disjoint S-Steiner trees 210.

Preliminaries

[0037] A *path* between two vertices u and v in a graph G (i.e., undirected graph 204) is a sequence $u = v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k = v$ such that e_i's are distinct edges of G, and the endpoints of e_i are v_{i-1} and v_i. Such a path is called *simple* if v_i's are all distinct. The *edge-connectivity* of a connected graph G is the minimum number k of edges required to remove from G to make it disconnected. Let $G(V, E)$ be a graph 204 and $S \subseteq V$ be a set of at least two vertices of G. For purposes of discussion, S is shown as a respective portion of "other data" 212. We say that S is k-edge-connected in G if for any set F of less than k edges in G, there is a path between any pair of vertices in S in $G \setminus F$. In other words, the S-

edge-connectivity of G is the minimum number of edges whose removal disconnects at least two vertices of S.

[0038] For a graph $G(V,E)$ 204, and a set $S \subseteq V$ of at least two vertices, an S-Steiner tree 210 is a subgraph $T(V',E')$ of G; $T(V',E')$ is a tree and $S \subseteq V'$. The *Steiner tree packing problem* for a given graph $G(V,E)$ 204 and $S \subseteq V$ is solved by the Steiner tree packing module 202, which finds a set of maximum size of edge-disjoint S-Steiner trees 210 of G 204 and combines them to generate the linked and disjoint packed (LED&P) Steiner trees 206, as now described.

Steiner Tree Packing

[0039] For a graph $G(V,E)$ and $S \subseteq V$ with $|S| = s$ (e.g., undirected graph 204), if S is k-edge-connected, then there are $\alpha_s k$ linked and edge-disjoint Steiner trees 206 for S, where α_s is a sequence that is asymptotic to $4/s$ as s tends to infinity (s tends to infinity is S). For simplicity of exposition, we first prove the following theorem, whose proof contains information that is later used to prove the immediately above.

[0040] *Theorem 2*: Let $G(V,E)$ be a graph 204 and $S = \{v_1, v_2, v_3\}$ be a subset of V. Assume that v_1 and v_2 are k-edge-connected, and v_1 and v_3 are $\frac{k}{2}$-edge-connected in G. Then G has $\frac{k}{2}$ edge-disjoint S- Steiner trees 210.

[0041] Fig. 3 shows an exemplary set of vertices of an input graph G 204. The exemplary vertices are used to demonstrate a shortcutting procedure to remove wasteful paths P from a Steiner tree, and thereby generate one or more edge-disjoint S-Steiner tree(s) 210. Referring to Fig. 3, the *Proof* of Theorem 2 follows.

[0042] Proof. Let

$$P = \{P_1, P_2, \ldots, P_{\frac{k}{2}}\}$$

be a set of $\frac{k}{2}$ edge-disjoint paths 208 between v_1 and v_3. Similarly, let $Q = \{Q_1, Q_2, \ldots, Q_k\}$ be a set of k paths 208 between v_1 and v_2. We consider the paths of P *from* v_3 *to* v_1; thus, an edge e appears *before* another edge e' on P_i if e is closer to v_3 than e' on P_i. Similarly, we consider the paths of Q from v_2 to v_1. Note that the paths in P are not necessarily disjoint from the paths in Q. The *last* intersection of a path Q_i with P is the last edge (i.e., the edge closest to v_1) on Q_i that is also part of a path in P. Assume that there exists a path $Q_i \in Q$ whose last intersection with P is $e \in P_j$ and e is not the last edge of P_j. Such a situation is called a *wasteful* situation, which may be removed via shortcutting operations.

<u>The Shortcutting Process</u>

[0043] A *shortcutting* procedure removes wasteful situations. For instance, and with respect to the immediately preceding example, a new path P_j^* is constructed by replacing the part of P_j *after* e with the part of Q_i after e. Notice that depending on whether e is traversed by P_j and Q_i in the same or opposite directions, e in P_j^* will or will not be included (P_j^* of Fig. 3 is marked with thick pixel lines as compared to the pixel thickness of other lines in the figure). Note that e is the last intersection of Q_i with P, therefore P_j^* does not intersect any path in $P \setminus P_j$. At this point, the shortcutting operation removes the shortcut path. In this example, P_j is removed from P and P_j^* is added to the LED&P Steiner tree 206. After doing this, P is still a set of $k/2$ edge-disjoint paths from v_3 to

v_1. We call this a *shortcutting* of P_j on Q_i, and say that Q_i is that path that is used for shortcutting P_j^*.

[0044] The shortcutting procedure is repeated as long as a wasteful situation is present in a Steiner tree that to generate an edge-disjoint Steiner tree 210 (path-tree). Let Q^* denote the set of paths in Q that are used for shortcutting a path in P (Q^* changes for each iteration of the algorithm). At any time, each path in P can be shortcut through at most one path in Q. Therefore, if P_j is first shortcut through Q_i (i.e., it is replaced by the path P_j^* constructed above) and at a later iteration, P_j^* is shortcut through $Q_{i'}$, then at this iteration the shortcutting operation removes i from Q^* and add i' instead. Let T denote the number of pairs (i, j) such that $P_j \in P$ and $Q_i \in Q$ intersect (T changes in each iteration of the algorithm).

[0045] In view of the above, the shortcutting procedure never increases the value of T. Additionally, each shortcutting iteration either increases the number paths in Q^*, or decreases T. Accordingly, it follows that the shortcutting operations eventually stop in a non-wasteful situation. In such a situation, every path $Q_i \in Q$ that has a non-empty intersection with P is used for shortcutting one path in P (i.e., belongs to Q^*). Therefore, the number of paths in Q that have an intersection with a path in P is not more than $k/2$. So if all edges of the paths in P from G are removed, there are still $\frac{k}{2}$ edge-disjoint paths from v_1 to v_2. Each such path together with a path from P forms an S-Steiner tree.

[0046] By induction on S and using the shortcutting procedure, for a graph $G(V,E)$ and $S \subseteq V$ where $S = \{v_1, v_2, \ldots, v_s\}$, if v_1 and v_i are $(i-1)k$-edge-connected

in G, for $2 \le i \le s$, then there are k edge-disjoint S-Steiner trees 206 in G. The existence of the same number of edge-disjoint S-Steiner trees 206 is substantially guaranteed under assumption that S is $(s-1)k$ -edge-connected, using the following simple argument: add a new vertex u and connect it to each of $v_2, v_3 ..., v_s$ with k parallel edges. In this graph, u and v_1 are $(s-1)k$ -edge-connected and therefore, by Menger's theorem, there are $(s-1)k$ edge-disjoint paths 208 from u to v_1. These paths can be partitioned into $s-1$ groups, such that the ith group consists of k paths between v_{i+1} and v_1. These paths are combined to obtain k edge-disjoint S-Steiner trees 206 in G.

[0047] These results substantially guarantee the existence of a collection of LED&P Steiner trees 206 all of which are *stars*. A star means that the sender is at the center and has edge disjoint paths going to all receivers.

[0048] *Theorem 3*: Let $G(V, E)$ be a graph 204 and S be a subset of s vertices of G. If S is k-edge-connected in G, then there are $\lfloor \alpha_s k \rfloor$ edge-disjoint S-Steiner trees in G 204, where α_i is defined by the following recurrence relation.

$$\alpha_2 = 1 \qquad \forall i > 2, \quad \alpha_i = \alpha_{i-1} - \alpha_{i-1}^2/4 \qquad (1).$$

Theorem 3 can be used to identity a better bound based on the proof of Theorem 2.

[0049] *Proof Sketch*: Induction on s is utilized. If $s = 2$, the theorem derives from Menger's theorem. Suppose $s > 2$, and let $v_1, v_2, ..., v_s$ be the vertices in S. Define $S' := S \setminus \{v_s\}$. By induction hypothesis, there is a collection T of $\alpha_{s-1}k$ edge-disjoint S'-Steiner trees 210 in G 204. We denote these Steiner trees

by $T_1,\ldots,T_{\alpha_{s-1}k}$. Also, from Menger's theorem, we know there are k edge-disjoint paths $P_1,P_2,\ldots,P_k$ 210 between v_s and v_1. We consider these paths 210 as paths starting from v_s. So, we say an edge e appears before (after) e' in P_j if e is closer (farther) to v_s than e'.

[0050] The basic idea is to combine these trees and paths to obtain LED&P Steiner trees 206 for S. A challenge arises when trees and paths have some edges in common (shared edge(s)). For purposes of discussion, an edge e is called a *red* edge if it is both in a tree T_i and a path P_j. Consider a tree, say T_1. For each required point v_i, $1 \le i \le s-1$, find the closest red edge to v_i in T_1. Let e be a red edge in T_1, i.e., e is in $T_1 \cap P_j$ for some path P_j. If e is the closest red edge to several vertices $v_{i_1},v_{i_2},\ldots,v_{i_t}$ in T_1, then we "shortcut" the path P_j to $v_{i_1},v_{i_2},\ldots,v_{i_t}$ at e. That is, we remove the part of P_j after e, and add to it the paths in T_1 between e and $v_{i_1},v_{i_2},\ldots,v_{i_t}$. Notice that after this operation, P_j is no longer a path; it starts from v_s as a path, but after reaching e it branches into several branches each ending in one of v_{i_r}'s. We call such a structure a *path-tree*, as we want to emphasize the distinction between the part between v_s and e (which comes from the original path P_j), and the part after e (that comes from the tree T_1).

[0051] Fig. 4 shows a set of nodes (e.g., network nodes) for showing the Proof of Theorem 3, which shows that there are a certain number of edge-disjoint S-Steiner trees in a graph as a function of a recurrence relation. Referring to Fig. 4, the collection of P_j's after the above shortcutting procedure are assumed to be edge-disjoint, since an intersection between P_j's can occur after the above procedure only if the paths between two required points v_i and v_j and their

respective closest red edges e_i and e_j intersect, and in such a situation we can pick e_i as the closest red edge to both v_i and v_j

[0052] After all paths that intersect T_1 in a red edge that is the closest red edge to one of v_i's are shortcut, the same shortcutting procedure is performed for for T_2. However, if a path P_j is shortcut at e while processing T_1, the edges of P_j that are discarded in this process (i.e., edges that come after e in P_j), are no longer considered red edges. We perform the shortcutting procedure on all trees $T_1,...,T_{\alpha_{s-1}k}$. For purposes of discussion, trees $T_1,...,T_{\alpha_{s-1}k}$ prior to being shortcut, are represented as S-Steiner trees 208.

[0053] After this process, paths may have been shortcut twice. For example, P_j might be shortcut at an edge e while processing T_1, and at an edge e' before e while processing T_2. If this happens, the edges of the part of P_j that is discarded during the shortcutting procedure for T_2 (i.e., the edges that come after e') are marked as non-red edges (so e is no longer a red edge), and the shortcutting procedure is again performed for T_1. That is, each of the vertices $v_{i_1}, v_{i_2},...,v_{i_t}$ that had e as their closest red edge in T_1 previously will have to choose their closest red edge again, with the updated set of red edges. This procedure iterates until every P_j is shortcut for at most one tree. It is apparent that this procedure ends, since every iteration discards some edges that were originally in P_j.

[0054] At this point, each P_j is shortcut for at most one tree T_i. Let Q_s denote the collection of P_j's that are not shortcut (and therefore are still paths from v_s to v_1), let f denote the size of Q_s. Also, let x_i ($i = 0,...,s-1$) denote the

number of T_j's that are used for shortcutting exactly i paths. For $i > 0$, from these trees and the paths that are shortcut through them, we get a collection of ix_i path-trees, that we denote by Q_i. We let Q_0 denote the collection of x_0 Steiner trees that are not used in shortcutting any path (ix_i path-trees and x_0 Steiner trees are represented as respective ones of the Edge-Disjoint S-Steiner trees 210).

[0055] From the above definitions, since there are $k - f$ of P_j's that are shortcut exactly once in the above procedure, we have

$$\sum_{i=0}^{s-1} ix_i = k - f.$$

For simplicity, we let $x_s := f/s$. Therefore, the above equation is written as

$$\sum_{i=0}^{s} ix_i = k. \qquad (2).$$

Also, since each of the T_j's is counted in exactly one of x_i's, we have

$$\sum_{i=0}^{s-1} x_i = \alpha_{s-1}k. \qquad (3).$$

The collection of all k paths and path-trees in $Q_1 \cup \cdots \cup Q_s$ and x_0 trees in Q_0 constitute a collection Q of $k + x_0$ edge-disjoint subgraphs (S-Steiner trees) 210 of G 204. In the rest of the proof, the subgraphs in Q are combined (merged) to construct LED&P S-Steiner trees 206 in G.

[0056] Let p be a number such that $\sum_{i=p+1}^{s} ix_i < x_0 \le \sum_{i=p}^{s} ix_i$. If $x_0 \le sx_s$, we define $p = s$, and if $\sum_{i=1}^{s} ix_i < x_0$, we define $p = 0$. For every $i = p+1,\ldots,s-1$, from each of the ix_i path-trees in Q_i, we pick one path from v_s to one of $v_1,\ldots,v_{s-1}$. Also, Q_s is by itself a collection of sx_s paths from v_s to v_1. Thus, we can obtain $\sum_{i=p+1}^{s} ix_i$ edge-disjoint paths from v_s to one of $v_1,\ldots,v_{s-1}$, from $Q_{p+1} \cup \cdots \cup Q_s$. There are px_p path-trees in Q_p, corresponding to x_p trees in T. Consider the path-trees corresponding to $\left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil$ of these trees,

and from each of these $p\left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil$ path-trees, take one path from v_s to one of $v_1,\ldots,v_{s-1}$. This gives us a collection of $p\left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil \geq x_0 - \sum_{i=p+1}^{s} ix_i$ paths from v_s to one of $v_1,\ldots,v_{s-1}$. Therefore, we get at least x_0 edge-disjoint paths from v_s to one of $v_1,\ldots,v_{s-1}$ at the expense of destroying the path-trees in $Q_s, Q_{s-1},\ldots,Q_{p+1}$, and the path-trees in Q_p corresponding to $\left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil$ trees in T. Each of these paths can be joined with one of the trees in Q_0 to form a LED&P S-Steiner tree 206. The remaining $px_p - p\left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil$ path-trees in Q_p can be grouped into $x_p - \left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil$ groups, each group consisting of p path-trees that correspond to the same tree in T. The union of the path-trees in each group is a graph that connects all vertices in S, and therefore contains an S-Steiner tree. This gives us $x_p - \left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil$ S-Steiner trees. Similarly, from each Q_i, $i = p, p-1,\ldots,1$, we get x_i S-Steiner trees. Therefore, the total number of LED&P S-Steiner trees 206 that we obtain is equal to

$$\begin{aligned} SOL_p(x) &= x_0 + x_p - \left\lceil (x_0 - \sum_{i=p+1}^{s} ix_i)/p \right\rceil + \sum_{i=1}^{p-1} x_i \\ &= \left\lfloor \frac{p-1}{p} x_0 + \sum_{i=1}^{p} x_i + \sum_{i=p+1}^{s} \frac{i}{p} x_i \right\rfloor \end{aligned} \qquad (4).$$

(Elements of SOLp(x) represents the worst case situation).

[0057] Here two special cases $p = 0$ and $p = s$ are considered. Using the same method, one can see that in these two cases $SOL_0(x) = \sum_{i=1}^{s} ix_i$ and $SOL_s(x) = \sum_{i=0}^{s-1} x_i$ S-Steiner trees, respectively, are produced. By Equations (2) and (3), we have $SOL_0(x) = k$ and $SOL_s(x) = \alpha_{s-1}k$. Therefore, in these cases we get at least $\alpha_{s-1}k > \alpha_s k$ edge-disjoint S-Steiner trees 206. Thus, we may assume without loss of generality that $1 \leq p < s$.

[0058] Now that we have computed the number of S-Steiner trees 206 produced in terms of x_i's, worst-case behavior is analyzed by treating x_i's as variables and solving the following linear program.

$$\begin{aligned} \text{minimize} \quad & \frac{p-1}{p}x_0 + \sum_{i=1}^{p} x_i + \sum_{i=p+1}^{s} \frac{i}{p}x_i \\ \text{subject to} \quad & \sum_{i=0}^{s} ix_i = k \\ & \sum_{i=0}^{s-1} x_i = \alpha_{s-1}k \\ & \forall i: \quad x_i \geq 0 \end{aligned} \qquad (5).$$

To upper bound the solution of the above linear program, we multiply its first constraint by $1/p^2$ and its second constraint by $(p-1)/p$. We obtain the following.

$$\sum_{i=0}^{s-1}\left(\frac{p-1}{p}+\frac{i}{p^2}\right)x_i + \frac{s}{p^2}x_s = \left(\frac{1}{p^2}+\frac{p-1}{p}\alpha_{s-1}\right)k \qquad (6).$$

In view of this, one can see that for $i \leq p$, $\frac{p-1}{p}+\frac{i}{p^2} \leq 1$ and for $i > p$, $\frac{p-1}{p}+\frac{i}{p^2} < \frac{i}{p}$. Also, $\frac{s}{p^2} < \frac{s}{p}$. Thus, since $x_i \geq 0$ for every i,

$$\begin{aligned} & \frac{p-1}{p}x_0 + \sum_{i=1}^{p} x_i + \sum_{i=p+1}^{s} \frac{i}{p}x_i \\ & \geq \sum_{i=0}^{s-1}\left(\frac{p-1}{p}+\frac{i}{p^2}\right)x_i + \frac{s}{p^2}x_s \\ & = \left(\frac{1}{p^2}+\frac{p-1}{p}\alpha_{s-1}\right)k \end{aligned} \qquad (7).$$

[0059] Equation (7) shows that in the worst case the Steiner tree packing module 202 of Fig. 2 finds at least $\lfloor (\frac{1}{p^2} + \frac{p-1}{p}\alpha_{s-1})k \rfloor$ edge-disjoint S-Steiner trees 206. The minimum of this expression is at $p = 2/\alpha_{s-1}$. Thus, the Steiner tree packing module 202 finds at least $\lfloor (\alpha_{s-1} - \alpha_{s-1}^2/4)k \rfloor = \lfloor \alpha_s k \rfloor$ edge-disjoint S-Steiner trees 206.

[0060] The algorithm given in the proof of Theorem 3 can be implemented in polynomial time. Polynomial time means that a run of the algorithm takes a small number of steps bounded above by a polynomial function of the graph size. Since the edge-connectivity of the set S is an upper bound on the maximum number of edge-disjoint S-Steiner 206 that can be packed in G, we arrive at the following:

[0061] *Corollary 1:* There is a polynomial time algorithm for the Steiner tree packing problem with an approximation ratio of α_s, where s is the number of required points.

[0062] *Lemma 1:* Let α_n be the sequence defined by Equation (1), the reoccurrence relation. Then $\alpha_n = \frac{4}{n} + o(\frac{1}{n})$.

[0063] *Proof.* Let $\beta_n = \frac{1}{2} - \frac{\alpha_n}{4}$, for $n \geq 2$. Therefore, from Equation (1) we have:

$$\beta_n = \beta_{n-1}^2 + \frac{1}{4}. \qquad (8).$$

[0064] Assertion *1*: For $n \geq 2$, $\beta_n \geq \frac{1}{2} - \frac{1}{n}$.

Proof. Induction is used on n. The statement holds trivially for $n = 2$. Suppose $n > 2$ and the claim is true for all values up to $n-1$. By Equation (8):

$$\beta_n \geq \left(\frac{1}{2} - \frac{1}{(n-1)}\right)^2 + \frac{1}{4}$$
$$= \frac{1}{2} - \frac{n-2}{(n-1)^2}$$
$$\geq \frac{1}{2} - \frac{1}{n}.$$

[0065] Assertion 2: For $n \geq 2$, $\beta_n \leq \frac{1}{2} - \frac{1}{4n}$.

Proof. Again, we use induction on n. The base case $n = 2$ is trivially true. Suppose the statement holds for all values up to $n-1$. By (8) and the induction hypothesis,

$$\beta_n \leq \left(\frac{1}{2} - \frac{1}{4(n-1)}\right)^2 + \frac{1}{4} = \frac{1}{2} - \frac{4n-5}{16(n-1)^2}$$
$$= \frac{1}{2} - \frac{4n^2 - 5n}{16n(n-1)^2} \leq \frac{1}{2} - \frac{4n^2 - 8n + 4}{16n(n-1)^2}$$
$$= \frac{1}{2} - \frac{1}{4n}.$$

This assertion is used to prove the following stronger statement/assertion.

[0066] Assertion 3: There is a constant c such that

$$\beta_n \leq \frac{1}{2} - \frac{1}{n} + \frac{c}{n \ln n}.$$

Proof. For small values of n, the claim is true if we let c to be a large enough constant. Let's assume that n is sufficiently large and that the claim is true for all integers up to $n-1$. From Equation (8) we have

$$\beta_n \leq \left(\frac{1}{2} - \frac{\ln(n-1) - c}{(n-1)\ln(n-1)}\right)^2 + \frac{1}{4}$$
$$= \frac{1}{2} - \frac{(\ln(n-1) - c)[(n-2)\ln(n-1) + c]}{(n-1)^2 \ln^2(n-1)}.$$

So, to prove the assertion, it is enough to show that,

$$\frac{(\ln(n-1)-c)[(n-2)\ln(n-1)+c]}{(n-1)^2\ln^2(n-1)} \geq \frac{\ln n - c}{n\ln n},$$

or equivalently,

$$\begin{aligned} &n\ln n(\ln(n-1)-c)[(n-2)\ln(n-1)+c] \\ &-(\ln n - c)(n-1)^2\ln^2(n-1)] \geq 0. \end{aligned} \qquad (9).$$

[0067] The expansion of the left-hand side of (9) is as follows:

$$\begin{aligned} &-cn^2\ln n\ln(n-1)+3cn\ln n\ln(n-1)-c^2n\ln n \\ &-\ln n\ln^2(n-1)+cn^2\ln^2(n-1) \\ &-2cn\ln^2(n-1)+c\ln^2(n-1) \\ &\geq cn^2\ln(n-1)[\ln(n-1)-\ln n]+cn\ln n\ln(n-1) \\ &-c^2n\ln n-\ln n\ln^2(n-1)+c\ln^2(n-1) \\ &\geq cn\ln(n-1)[\ln n-2]-c^2n\ln n \\ &-\ln n\ln^2(n-1). \end{aligned}$$

Let $c = \frac{3}{4}\ln n_0$ and let n_0 be the smallest integer such that

$$\begin{aligned} &cn_0\ln(n_0-1)[\ln n_0-2]-c^2n_0\ln n_0 \\ &-\ln n_0\ln^2(n_0-1)\geq 0. \end{aligned}$$

By this definition, assertion (3) is true for $n \leq n_0$ by assertion 2, and for $n > n_0$ by Equation (10).

[0068] From assertions 1 and 3,

$$\frac{4}{n}-\frac{4c}{n\ln n} \leq \alpha_n \leq \frac{4}{n}.$$

This completes the proof of the lemma.

[0069] The simplest case of the Steiner tree packing problem for solution by the Steiner tree packing module 202 of Fig. 2 is where the number of requisite points is three ($|S|=3$). In such a scenario, Theorem 3 provides the following:

Let $G(V,E)$ be a graph and S be a subset of 3 vertices of G. If S is k-edge-connected in G, then there are $\lfloor \frac{3}{4}k \rfloor$ edge-disjoint S-Steiner trees in G. The following example shows that the constant 3/4 in the above corollary is not replaced with any larger constant.

[0070] Let G be a graph on s vertices with exactly r parallel edges between each pair of its vertices, and let $S = V(G)$. Clearly, S is k-edge-connected, where $k = (s-1)r$. Since each S-Steiner tree has exactly $s-1$ edges, the maximum number of edge-disjoint S-Steiner trees in G is at most $r\binom{s}{2}/(s-1) = rs/2 = \frac{s}{2(s-1)}k$. In particular, when $s = 3$, the graph does not contain more than $\frac{3}{4}k$ edge-disjoint Steiner trees.

An Exemplary Procedure

[0071] Fig. 5 shows an exemplary procedure 500 to locate a substantially maximum number of LED&P S-Steiner trees 206 (Fig. 2) that can be used in many practical applications to connect to a same set S of terminal vertices of a graph 204 (Fig. 2). For instance, with respect to data broadcasting (network data communication), a data broadcaster will typically desire to broadcast as many streams of data (e.g., movies) to end users (receivers) as possible. This is to substantially optimize data throughput and to provide end-users with a large number of data viewing options. If parallel edges are allowed, each link from the sender node to a receiver node may carry only one data stream. Parallel edges mean two edges between the same two nodes. If an edge has twice the capacity then a regular edge than that edge may be represented by two parallel edges. So we may assume that all the edges have the same capacity. In this scenario, to

broadcast as many streams of data (e.g., movies) to end users (receivers) as substantially possible, the data broadcaster (sender) utilizes the Steiner tree packing module 202 of this exemplary procedure to analyze network node information 204 and produce/find a substantially maximum number of edge-disjoint Steiner trees 206 connecting the broadcaster to all users (receiver nodes). Subsequently, the broadcaster multicasts each data stream (e.g., of a movie) to the receiver(s) via the produced Steiner tree(s) 206.

[0072] In another example with respect to a VLSI implementation of the Steiner tree packing module 202, the packed Steiner tree(s) 206 are used to share an electric signal by a set of terminal nodes.

[0073] For purposes of discussion, the operations of procedure 500 are described in reference to the features of Figs. 1 and 2. The left-most digit of a component reference number identifies the particular figure in which the component first appears. In this implementation, the procedure 500 generates LED&P S-Steiner trees 206 (Fig. 2) for multicast of streaming media from a broadcast terminal (network node) to one or more receiving terminals (network nodes). In this implementation, the computing device 130 (Fig. 1) represents a sender node (data broadcasting node), and respective implementations of the remote computing device 182 (Fig. 1) represent receiver and router (network switching, Steiner) nodes. Graph 204 is an undirected connected finite graph that provides sender, receiver, and router network node (vertex) input data. Data for streaming to the one or more receiver nodes is shown as a respective portion of "other data" 212.

[0074] At block 502, the Steiner tree packing module 202 (Fig. 2) analyzes the undirected graph 204 (input data) to produce a first Steiner tree 210 between two terminals, a sender and a receiver terminal. In this implementation, this is accomplished via Menger's theorem, which was described above. At block 504, the Steiner tree packing module 202 gets a next data point from the undirected finite graph 204. The data point, at this juncture, is a receiver node. An unprocessed data point is a data point that has not been connected to at least one of the Steiner tree(s) 210.

[0075] At block 506, the Steiner tree packing module 202 inductively grows the graph 204 with respect to this new data point. More specifically, the Steiner tree packing module 202 identifies one or more disjoint paths 208 from the new data point to the data terminals via any proceed Steiner nodes already connected to the processed nodes. At block 508, the Steiner tree packing module 202 determines if all data points in the graph 204 have been processed. If not, the procedure 500 continues at block 504 as described above. Otherwise, the procedure continues at block 510. At block 510, the Steiner tree packing module 202 merges the generated disjoint Steiner trees 210 with the disjoint paths 208 via the described shortcutting procedure while removing as many wasteful paths as substantially possible. This operation generates the LED&P Steiner trees 206.

[0076] For purposes of illustration, at block 512, the LED&P Steiner trees 206 are used by an application. For instance, the data broadcasting module of "Other Program Modules" 162 utilizes the LED&P Steiner trees 206 to

substantially optimize its response to streaming data requesting network nodes. In this implementation, this is accomplished by multicasting streaming data to the set *S* of requesting terminals that are identified in at least one S-Steiner tree 206–the number of trees being selected by the application as a function of desired data throughput

Conclusion

[0077] The described systems and methods produce Steiner trees identifying a substantial maximum number of linked, edge-disjoint, and packed Steiner trees 206. Although the systems and methods have been described in language specific to structural features and methodological operations, the subject matter as defined in the appended claims are not necessarily limited to the specific features or operations described. For instance, although systems and methods of the invention have been described in reference to packing Steiner trees for multicast of streaming media, the systems and methods can also be utilized in other practical applications, such as in VLSI circuit design. Accordingly, the specific features and operations are disclosed as exemplary forms of implementing the claimed subject matter.

References

[0078] The following references [1] through [3] are hereby incorporated by reference.

[1] M. Kriesell. Local Spanning Tres in graphs and hypergraph decomposition with respect to edge connectivity. Technical Report 257, University of Hanover, 1999.

[2] A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into *k*-connected sub-hypergraphs. Technical report published by the Egreváry Research Group, Budapest, Hungary. ISSN 1587-4451, 2001.

[3] L. Petingi and J. Rodriguez. Bounds on the Maximum Number of Edge-disjoint Steiner Trees of a Graph. Congresus Numerantium, 145:43-52, 2000.